Procedia Computer Science

Volume 51, 2015, Pages 630–639

ICCS 2015 International Conference On Computational Science

Parallel metaheuristics in computational biology: an asynchronous cooperative enhanced Scatter Search method

David R. Penas[1], Patricia González[2], José A. Egea[3], Julio R. Banga[1], and Ramón Doallo[2]

[1] BioProcess Engineering Group
IIM-CSIC - Instituto de Investigaciones Marinas
C/Eduardo Cabello, 6. E-36208 Vigo, Spain
{davidrodpenas,julio}@iim.csic.es
[2] Computer Architecture Group
University of A Coruña
Campus de Elviña s/n. 15071 A Coruña, Spain
{patricia.gonzalez,doallo}@udc.es
[3] Department of Applied Mathematics and Statistics
Universidad Politécnica de Cartagena
Avenida Dr. Fleming s/n, 30202 Cartagena, Spain
josea.egea@upct.es

Abstract

Metaheuristics are gaining increased attention as efficient solvers for hard global optimization problems arising in bioinformatics and computational systems biology. Scatter Search (SS) is one of the recent outstanding algorithms in that class. However, its application to very hard problems, like those considering parameter estimation in dynamic models of systems biology, still results in excessive computation times. In order to reduce the computational cost of the SS and improve its success, several research efforts have been made to propose different variants of the algorithm, including parallel approaches.

This work presents an asynchronous Cooperative enhanced Scatter Search (aCeSS) based on the parallel execution of different enhanced Scatter Search threads and the cooperation between them. The main features of the proposed solution are: low overhead in the cooperation step, by means of an asynchronous protocol to exchange information between processes; more effectiveness of the cooperation step, since the exchange of information is driven by quality of the solution obtained in each process, rather than by a time elapsed; optimal use of available resources, thanks to a complete distributed approach that avoids idle processes at any moment. Several challenging parameter estimation problems from the domain of computational systems biology are used to assess the efficiency of the proposal and evaluate its scalability in a parallel environment.

Keywords: Computational Systems Biology, Parallel Metaheuristics, Enhanced Scatter Search

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2015

doi:10.1016/j.procs.2015.05.331

1 Introduction

The aim of systems biology is to generate new knowledge about complex biological systems by combining experimental data with mathematical modeling and advanced computational techniques. Dynamic models (described by sets of nonlinear ordinary differential equations) is the most common formalism in this area. Model calibration consists of finding the parameters of a dynamic model that give the best fit to a set of time-series experimental data, which entails minimizing a cost function that measures the goodness of this fit. This class of problems is extremely challeging due to its ill-conditioned and multimodal nature [9]. To efficiently solve the calibration problem many research efforts have focused on developing metaheuristic methods, which combine mechanisms for exploring the search space and exploiting previous obtained knowdlege. Those mechanisms make the methods be able to find good solutions in reasonable computation times [2].

Scatter search (SS) [6] is a population-based method that has recently been demonstrated to produce promising results for solving combinatorial and nonlinear optimization problems. It uses strategies for combining solution vectors that have proved effective in a variety of problem settings. Recently, a cooperative enhanced Scatter Search (CeSS) has been presented in [10]. This work demonstrates that the cooperation of individual parallel searches modifies the systemic properties of the individual algorithms, improving its performance and outperforming other competitive methods. However, the parallel strategy followed presents serious issues in terms of computational efficiency and scalability.

This paper presents an asynchronous cooperative strategy to implement a parallel enhanced Scatter Search algorithm. Building upon the ideas outlined in [10], this work makes the following major contributions:

- an asynchronous communication protocol to handle inter-process information exchange, avoiding idle processes while waiting for information exchanged from other processes
- an exchange of information driven by quality of the solutions obtained by each individual process, rather than by time elapsed, to achieve more effective cooperation between processes
- a distributed approach, instead of the classical centralized master-slave solutions, to improve the use of the available resources.

The structure of this paper is as follows. Section 2 briefly presents the background and related work. Section 3 presents the proposal for an asynchronous cooperative parallel Scatter Search. Section 4 describes the experiments carried out and discusses on the obtained results. Finally, Section 5 summarizes the conclusions of the paper and the future work.

2 Background

Scatter search (SS) [6], from the point of view of metaheuristic classification, is a population-based algorithm that constructs solutions by applying strategies of diversification, improvement, combination and population update. It derives its foundations from strategies originally proposed for combining decision rules and constraints in the context of integer programming. The goal of this methodology is to enable the implementation of solution strategies that can derive new solutions from combined elements in order to produce better solutions than those strategies that base their combinations only on a set of original elements.

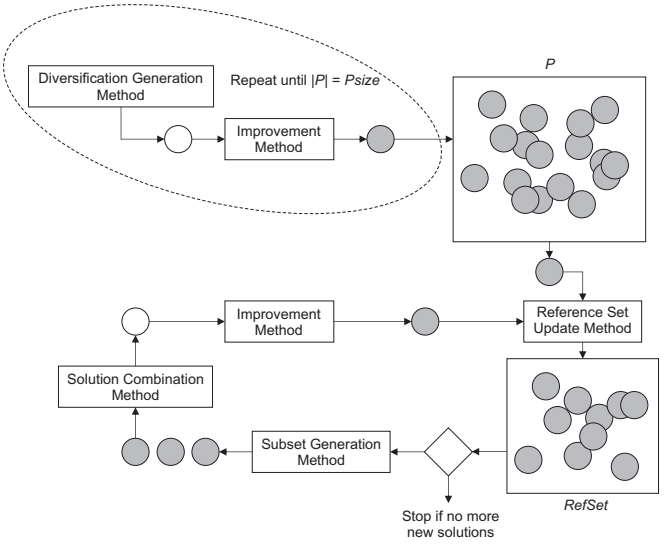


Figure 1: Schematic representation of a basic Scatter Search

Compared with other methods, like genetic algorithms, Scatter Search makes use of a low number of population members, called *Reference Set*. Besides, it includes an *improvement method* that usually consists in a local search to speed-up the convergence to optimal solutions.

The method starts by generating an initial population of solutions within the search space. A good representative solutions of the population are selected to create the initial reference set. Then a subset is selected from this reference set, and its solutions are combined to obtain new ones which are candidate starting solutions in a subsequent improvement procedure. After the mentioned mechanisms, a reference set update method is applied to create the population of the next iteration before the procedure is repeated until the end of the search. Thus, the SS strategy involves 5 procedures: the initial population creation method, the reference set generation method, the subset generation method, the solution combination method and the improvement solution method. Figure 1 shows a schematic representation of a basic SS design.

In [4, 5] an enhanced Scatter Search (eSS) method for the global dynamic optimization of nonlinear processes is presented. This new method presents a simpler but more effective design which helps to overcome typical difficulties of nonlinear dynamic systems optimization such as noise, flat areas, nonsmoothness, and/or discontinuities. A basic pseudocode of the eSS algorithm is shown in Algorithm 1. Innovative mechanisms are implemented in different parts of the method that address the balance between intensification (local search) and diversification (global search):

- It uses a small population size, even for large-scale problems, but allowing more search directions than in classical SS thanks to a new combination scheme. The number of evaluations per iteration does not increase while the diversity in the search is preserved.

- It implements an intensification mechanism in the global phase, which exploits the promising directions defined by a pair of solutions in the reference set.

- Besides, a heuristic local search method is used to accelerate the convergence of large-scale problems.

Algorithm 1: Enhanced Scatter Search algorithm - `eSS`

```
Create_Population(ndiverse);
Generate_RefSet(RefSet, dimRefSet);
repeat
    Sort_RefSet(RefSet);
    for i=1 to dimRefSet do
        Combine_Solutions(RefSet, NewSol);
        Improve_Solutions(NewSol, ImprSol);
    end
    Update_RefSet(RefSet);
until stopping criterion;
```

- It also implements diversification strategies that make use of memory (introducing a *tabu list*) to infer whether a solution is stagnated in a local optimum or if it is too close to previously found solutions.

Thus, the eSS method provides a good balance between robustness and efficiency in the global phase, and couples a local search procedure to accelerate the convergence to optimal solutions. However, for most hard problems, like those considering parameter estimation in dynamic models, eSS still requires excessive computation times.

Recently, in [10], a cooperative parallel strategy for the eSS method has been presented. The proposed Cooperative enhanced Scatter Search (CeSS) follows a master-slave approach, where each of the *slave* processors runs a sequential eSS algorithm, while they exchange their reference set of solutions through a *master* processor at certain fixed instants. The results presented in [10] show how cooperation of individual parallel search *slaves* modifies the systemic properties of the individual algorithms, improving its performance and outperforming other competitive methods. However, the parallel strategy followed presents serious issues. First, exchanging information at fixed instants, in a blocking fashion, forces a synchronization that delays the progress in many slaves. Processors are idle during a significant amount of time, while they are waiting for each other during the migration steps. The penalty may be significant because the improvements introduced by the eSS in diversification, like the local solver or the use of a tabu list, lead processes into a more computationally unbalanced scenario. This synchronization is one of the causes of its poor scalability when the number of processors grows. Another drawback of this strategy is that the *master* processor is sit idle between the communication phases.

3 Asynchronous Cooperative enhanced Scatter Search

The parallelization of metaheuristics pursues one or more of the following goals: increase the size of the problems that can be solved, speed-up the computations, or attempt a more thorough exploration of the solution space [3, 1]. However, achieving an efficient parallelization of metaheuristics is usually a complex task, since the search of new solutions depends on previous iterations of the algorithm, which not only complicates the parallelization itself but also limits the speedup achievable.

The solution explored in this work pursues the development of an efficient asynchronous

cooperative enhanced Scatter Search, focussed on both the acceleration of the computation by performing separate evaluations in parallel, and the convergence improvement through the stimulation of the diversification in the search and the cooperation between different processes.

The parallel algorithm proposed is based on the cooperation between parallel processes, by means of an exchange of information driven by the quality of the solutions obtained in each individual process. For cooperation to be efficient, in large-scale difficult problems, like calibration in large-scale systems biology models, each parallel process must adopt a different strategy to increase the diversification in the search. The idea is to run in parallel processes with different degree of *agressiveness*. Some processes will focus on diversification (global search) increasing the probabilities of finding a feasible solution even in a *rough* or difficult space. Other processes will concentrate in intensification (local search) and speed-up the computations in *smoother* spaces. Cooperation between them enables each process to benefit from the knowledge gathered by the rest.

However, the implementation of the cooperation step is not as straightforward as it may appear to be. The naive solution would be to implement a communication protocol where all the processes communicates with the rest, i.e. an all-to-all communication step. However, this approach will become a bottleneck, specially when the number of processes grows, leading to a solution that would difficultly scale. Also the convergence of the cooperative algorithm will be altered, since most of the communications will arrive delayed, thus being useless for many processes that already have reached better solutions by themselves. To alleviate the communications in the cooperation step and allow for the execution progress, the proposed communication strategy uses a ring topology. Actually, a double-direction ring topology is proposed, that is, each process i in the ring will exchange information with both its previous neighbor $(i-1)$ and its next neighbor $(i+1)$.

The pseudocode for the asynchronous Cooperative enhanced Scatter Search (aCeSS) is shown in Algorithm 2. At the begining of the algorithm, a local variable in each process is declared to keep track of the best solution shared in the cooperation step. Each process creates its own population matrix of *ndiverse* solutions, where *ndiverse* is different for different processes depending on their agressiveness. Then a initial *RefSet* is generated for each process with *dimRefSet* solutions with high quality and random elements. Again different *dimRefSet* are possible for different processes. The rest of the operations are performed within each *RefSet* in each process, in the same way as in the serial eSS implementation. Every external iteration of the algorithm, a cooperation phase is performed to exchange information with the rest of the processes in the parallel application. Whenever a process reaches the cooperation phase, it checks if any message with a new best solution from the double-direction ring has arrived to its reception memory buffers. If a new solution has arrived, the process checks whether this new solution improves the current shared best solution or not. If the new solution improves the current one, the new solution promotes to be the best solution shared with this process till now. The loop to check the reception of new solutions must be repeated until there are no more shared solutions to attend. This is because the execution time of one external iteration may be very different from one process to another, due to the diversification strategy explained before. Thus, while a process has complete only one external iteration, their neighbors may have complete more, and several messages may be waiting in the reception buffer. The best solution shared till now replaces then the worst solution in the process *RefSet*. After the reception step, the process checks whether its best solution improves in, at least, an ϵ the best solution shared. If its best solution improves the shared one, it updates the shared best solution with its best solution. Finally, if the shared best solution has changed in this iteration, the process sends it to their neighbors in the ring. The algorithm repeats the external loop until the stopping

Algorithm 2: Asynchronous Cooperative enhanced Scatter Search algorithm - `aCeSS`

```
coopBestSol = DBL_MAX;
Create_Population(ndiverse);
Generate_RefSet(RefSet, dimRefSet);
repeat
    ! Serial eSS
    Sort_RefSet(RefSet);
    for i=1 to dimRefSet do
        Combine_Solutions(RefSet, NewSol);
        Improve_Solutions(NewSol, ImprSol);
    end
    Update_RefSet(RefSet);

    ! Cooperation step
    sendflag=false;
    recvflag=true;
    while recvflag do
        Non_Blocking_Recv(RecvSol,(preN,posN),recvflag);
        if RecvSol < coopBestSol then
            coopBestSol=RecvSol;
            sendflag=true;
        end
    end
    if sendflag then
        Replace_Worst_Solution(worstSol,coopBestSol);
    end
    if ((coopBestSol − bestSol)/bestSol) < ϵ then
        coopBestSol=bestSol;
        sendflag=true;
    end
    if sendflag then
        Non_Blocking_Send(coopBestSol,(preN,posN));
    end
until stopping criterion;
```

criterion is met.

Note that both the sends and receptions of the messages are performed using non-blocking operations, thus allowing for the overlap of communications and computations. Besides, the proposed communication protocol avoids process stalls if messages have not arrived during an external iteration, allowing for the progress of the execution in every individual process. This is crucial in the application of the aCeSS method to solve large-scale difficult problems, since the algorithm success heavily depends on the diversification degree introduced in the different processes, that will result in them running asynchronously and a computationally unbalanced scenario. To implement the proposed protocol, the non-blocking point-to-point communication functions of the Message-Passing Interface (MPI) standard have been used [8].

4 Experimental results

In order to evaluate the proposed cooperative asynchronous algorithm (aCeSS), the benchmarks from the BioPreDyn-bench suite [11] have been tested. These are challenging parameter estimation problems from the domain of computational system biology. Results for three of these benchmarks are reported in this section:

- *Problem B2*: dynamic model of the Central Carbon Metabolism of *E. coli*. It consists of 116 kinetic parameters and maximum reaction rates to be estimated.
- *Problem B4*: metabolic model of Chinese Hamster Ovary (CHO) cells. It consists of an ODE model comprising 117 parameters in total.
- *Problem B5*: signal transduction logic model. It consists of 26 ODEs that use a logic-based formalism and include 86 continuous parameters.

To assess the efficiency of the aCeSS method, different experiments have been carried out. Its behavior, in terms of convergence and total execution time, was compared with the sequential version of the enhanced Scatter Search (eSS) [4] and its synchronous cooperative parallel version (CeSS) [10]. Although the reported implementations of the eSS and CeSS were coded in Matlab, to perform here a fair comparison both algorithms have been implemented in F90. The original CeSS implementation used *jPar* [7] to perform the parallel executions and the communication between master and slaves. In the new developed version, to compare with the aCeSS, the MPI library has been employed.

A multicore cluster was used to carry out these experiments. It consists of 16 nodes powered by two octa-core Intel Xeon E5-2660 CPUs with 64 GB of RAM. The cluster nodes are connected through an InfiniBand FDR network. OpenMPI library version 1.6.2 has been used to compile the parallel implementations, and the experiments were carried out using from 1 to 30 processors.

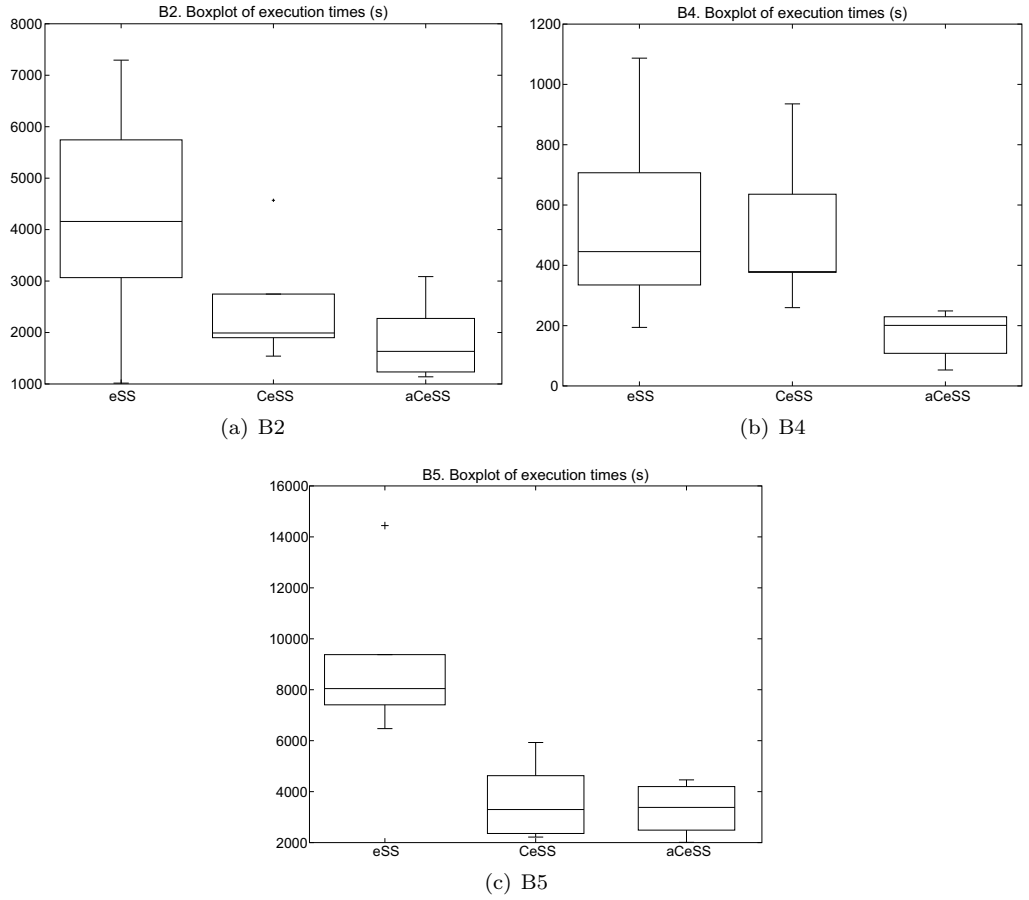
With the aim of performing the most honest comparison, the tested scenario was the following: for each experiment, serial or cooperative, 10 processors were available for computing. This means that, when executing the serial algorithm, 10 processors are available to run 10 eSS algorithms in parallel (without cooperation). In order to perform a fair comparison, diversity is introduced in these 10 eSS runs in the same sense as CeSS and aCeSS do, i.e. each one performing a different eSS with a different strategy.

Comparing the sequential and the cooperative metaheuristic is not an easy task, due to the substantial dispersion of experimental results in these stochastic problems. For this experimental evaluation, the cooperative metaheuristics CeSS and aCeSS were executed until their achieved solutions had a similar accuracy as the serial eSS, i.e., they are compared based on a quality solution. The value-to-reach (VTR) used was the optimal fitness value reported in [11]. Each experiment was performed at least 5 times.

Average mean execution times and speedup results for these experiments are shown in Table 1. This table displays, for each experiment, the average number of evaluations needed to achieve the VTR (*# evals*), the mean, minimum, and maximum execution time of all the runs in the experiment, and the speedup achieved. The speedup may appear to be very modest, however, note that the speedups were calculated comparing execution times of aCeSS and CeSS with the execution time of eSS, using in all the cases 10 processors (also in the eSS method as explained above). In these experiments the computational load is not shared among processors, thus, the total speedup achieved over the eSS depends on the impact that the cooperation

P	method	#evals	time(s) mean	min.	max.	speedup
	eSS	104598	4256.10	1016.46	7293.75	-
B2	CeSS	87031	2550.05	1541.85	4569.63	1.67
	aCeSS	57005	1873.45	1139.12	3086.56	2.27
	eSS	18195	544.37	194.08	1087.07	-
B4	CeSS	16333	517.33	259.63	935.34	1.05
	aCeSS	2427	168.29	52.97	248.82	3.23
	eSS	19575	9149.24	6475.00	14442.85	-
B5	CeSS	7992	3686.18	2217.88	5929.51	2.48
	aCeSS	7443	3308.73	2010.70	4462.15	2.77

Table 1: Execution time and speedup results using 10 processors. $VTR_{B2} = 2.1051 \times 10^2$, $VTR_{B4} = 4.5718 \times 10^{-1}$ and $VTR_{B5} = 3.0725 \times 10^3$.



(a) B2 (b) B4

(c) B5

Figure 2: Box plot of the execution times.

among processes produces on achieving a good result performing less number of evaluations and hence better performance. These figures show that the proposed aCeSS method reduces the number of evaluations and the execution time of the eSS and CeSS.

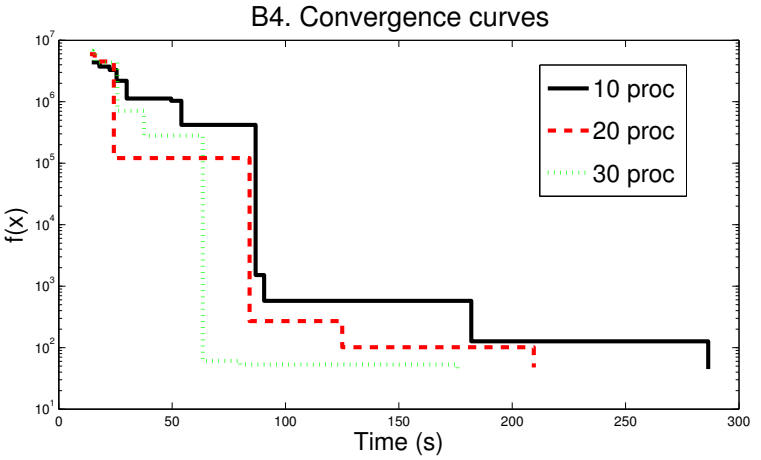


Figure 3: Convergence curves for different number of processors.

In this kind of stochastic problems it is important to evaluate the dispersion of the experimental results. Figure 2 illustrates how the proposed aCeSS reduces the variability of execution time in the eSS and the CeSS method. It demonstrates that the outliers of the execution time decrease in the asynchronous cooperative method. This is an important feature of the aCeSS, because it reduces the average execution time for each benchmark.

One known issue of the CeSS method is its poor scalability, since its synchronous communication step is challenged by the increasingly computationally unbalanced scenario when the number of processors grows. In order to evaluate the scalability of the aCeSS, it has been tested using 10, 20 and 30 processors. Figure 3 shows, for B4 benchmark, the convergence curves for those experiments that fall in the median values of the results distribution. Similar results are obtained for B2 and B5 benchmarks. It can be seen that the aCeSS still improves the convergence results when the number of processors grows, thanks to the asynchronous communication protocol.

5 Conclusions

In this paper an asynchronous cooperative parallel strategy for the enhanced Scatter Search algorithm (aCeSS) is presented. The designed approach is an effective way to cooperate between different processes running different configurations of the eSS algorithm. Three are the main features of the proposed approach: (i) it is a completely distributed approach, differing from the most popular centralized master-slave cooperative approaches; (ii) the exchange of information between cooperating processes is driven by the quality of the solution encountered in each individual process, rather than by a time elapsed; (iii) the cooperation phase is based on an asynchronous communication protocol, that avoids processes' stalls and allows for execution progress in a computational unbalanced scenario.

The testbed used to evaluate the proposed asynchronous Cooperative enhanced Scatter Search (aCeSS) was comprised of three challenging parameter estimation problems from the domain of computational systems biology. The experimental results show that the asynchronous parallel strategy proposed attains a reduction in the convergence time through cooperation of the parallel processes, demonstrating also a competitive speedup against a synchronous

cooperative strategy (CeSS).

Despite the reduction in the convergence time, the diverse eSS methods running in different processors in the parallel execution cause often situations where only some processes, the most promising ones, are able to share solutions with the rest. This limits the scalability of the proposal, since those processes that never obtain solutions to be shared, can be considered idle processes. Therefore, future work will focus on developing auto-tuning heuristics that allow for reconfiguring the slowest processes with the successful parameters of the promising processes.

Acknowledgment

This research received financial support from the Spanish Ministerio de Economía y Competitividad (and the FEDER) through the projects DPI2011-28112-C04-03, DPI2011-28112-C04-04, and TIN2013-42148-P, from the CSIC intramural project "BioREDES" (PIE-201170E018) and from the Galician Government under the Consolidation Program of Competitive Research Units (Network ref. R2014/041 and competitive reference groups GRC2013/055). D. R. Penas acknowledges financial support from the MICINN-FPI programme.

References

[1] Enrique Alba. *Parallel metaheuristics: a new class of algorithms*, volume 47. Wiley-Interscience, 2005.

[2] Julio R Banga. Optimization in computational systems biology. *BMC Systems Biology*, 2(1):47, 2008.

[3] Teodor Gabriel Crainic and Michel Toulouse. *Parallel strategies for meta-heuristics*. Springer, 2003.

[4] Jose A Egea, Eva Balsa-Canto, Maria S G García, and Julio R Banga. Dynamic optimization of nonlinear processes with an enhanced scatter search method. *Industrial & Engineering Chemistry Research*, 48(9):4388–4401, 2009.

[5] Jose A Egea, Rafael Martí, and Julio R Banga. An evolutionary method for complex-process optimization. *Computers & Operations Research*, 37(2):315–324, 2010.

[6] Fred Glover, Manuel Laguna, and Rafael Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.

[7] A Karbowski, M Majchrowski, and P Trojanek. jPar–a simple, free and lightweight tool for parallelizing Matlab calculations on multicores and in clusters. In *9th International Workshop on State-of-the-Art in Scientific and Parallel Computing (PARA 2008)*, 2008.

[8] Peter S. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[9] Alejandro F Villaverde and Julio R Banga. Reverse engineering and identification in systems biology: strategies, perspectives and challenges. *Journal of The Royal Society Interface*, 11(91):20130505, 2014.

[10] Alejandro F Villaverde, Jose A Egea, and Julio R Banga. A cooperative strategy for parameter estimation in large scale systems biology models. *BMC Systems Biology*, 6(1):75, 2012.

[11] Alejandro F Villaverde, David Henriques, Kieran Smallbone, Sophia Bongard, Joachim Schmid, Damjan Cicin-Sain, Anton Crombach, Julio Saez-Rodriguez, Klaus Mauch, Eva Balsa-Canto, Pedro Mendes, Johannes Jaeger, and Julio R Banga. Biopredyn-bench: a suite of benchmark problems for dynamic modelling in systems biology. *BMC Systems Biology*, 2015. In press.